OULUN AMMATTIKORKEAKOULU

Justus Pousi

Unity-moninpeli HLAPI-rajapinnalla

Unity-moninpeli HLAPI-rajapinnalla

Justus Pousi
Opinnäytetyö
Syksy 2016
Tietojenkäsittelyn koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma, web-sovellusten kehittäminen

---

Tekijä: Justus Pousi
Opinnäytetyön nimi: Unity-moninpeli HLAPI-rajapinnalla
Työn ohjaaja: Matti Viitala
Työn valmistumislukukausi- ja vuosi: Syksy 2016 Sivumäärä: 29

Tässä opinnäytetyössä tutustutaan Unity-pelinkehitysalustan HLAPI-rajapintaan ja kerrotaan kokemuksista tehdä yksinkertainen verkkopeli niiden avulla. Teoriaosuudessa käydään lyhyesti läpi käytettyjä työkaluja sekä Unityn HLAPI-rajapinnan tärkeimpiä ominaisuuksia. Opinnäytetyöllä ei ollut toimeksiantajaa, vaan tein sen omasta kiinnostuksesta aiheeseen ja oppiakseni enemmän verkkopelien kehittämisestä.

Pelissä pelaajat ajavat leijuvaa hovercraft-ajoneuvoa laajassa proseduurillisesti luodussa aavikkokentässä. Tavoitteena on kerätä mahdollisimman paljon kentällä olevia sinisiä palloja jotka antavat polttoainetta ja pisteitä pelaajalle.

Opinnäytetyön tavoitteet saavutettiin ja tuotoksena oli pelin prototyyppi, jossa on toimivat verkko-ominaisuudet. Unityn HLAPI-rajapinta todettiin pääosin helposti käytettäväksi ratkaisuksi verkkopelien tekemiseen. Pelin prototyypillä on jatkokehitysmahdollisuuksia esimerkiksi pelin tavoitteiden uudelleensuunnittelu ja maaston luonnin parantaminen.

Asiasanat: Unity, ohjelmointi, verkkopeli, pelinkehitys

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems, Web Application Development

---

Author: Justus Pousi
Title of thesis: Unity multiplayer game with HLAPI
Supervisor: Matti Viitala
Term and year when the thesis was submitted: Autumn 2016 Number of pages: 29

The main goals of this thesis are first to introduce the High Level API (HLAPI) of the Unity game engine and second to show how I developed an online game prototype with HLAPI. In the theoretical part of the thesis I describe the tools I used together with the most important features of HLAPI. The thesis was not commissioned by anyone, but was rather done out of my own interest in the subject to learn more about making online games.

In the game, players control a flying hovercraft on a large procedurally generated desert level. On the level there are many blue spheres that yield points and fuel. The aim is to collect as many of these as possible.

The goals of the thesis were achieved and the game that was produced has working online capabilities. HLAPI was found to be an approachable option for making online games. The game prototype has possibilities for continued development, for example, redesigning the goals of the game and remaking the terrain generation.

Keywords: Unity, programming, online game, game development

SISÄLLYS

1 JOHDANTO

Verkkopelien eli internetin välityksellä pelattavien videopelien määrä ja suosio on kasvanut räjähdysmäisesti 2000-luvulla. Samalla pelien kehittäminen on helpottunut huomattavasti useiden ilmaisten pelikehitysympäristöjen julkaisun myötä. Yksi suosittu pelikehitysympäristö on Unity, johon lisättiin 2015 julkaistussa Unity 5.1 versiossa paremman tuen luoda verkkopelejä ”The High Level API” (HLAPI) -rajapinnalla (Lian 2015, viitattu 31.8.2016). Tässä opinnäytetyössä käydään läpi HLAPI:n yleisimpiä ominaisuuksia sekä kerron kokemuksistani tehdä verkkopelin prototyyppi Unityllä.

Verkkopelien verkkojärjestelmät ovat yksi laajimmista alueista pelinkehityksessä. Niitä myös pidetään yhtenä teknisesti haastavimmista alueista peliohjelmoinnista. Moninpeli on osana pelisuunnittelun ydintä useissa peleissä, ja verkkoyhteyksiä käytetään myös monissa yksinpeleissä. Tästä syystä pelien verkkojärjestelmien opettelu on hyvä taito opetella kelle tahansa ohjelmoijalle, joka hakee työmahdollisuuksia pelikehityksessä.

Suunnitelmana oli tehdä verkkopelin prototyyppi käyttäen Unityn HLAPI -rajapintaa jotta saisin kokemusta sen käyttämisestä. Pelin idea on hyvin yksinkertainen. Pelissä kerätään pisteitä ajamalla leijuvaa moottoripyörän tyyppistä ajoneuvoa kuvitteellisessa aavikko areenassa. Pelinkehitys tapahtui suurilta osin iteratiivisella kehitystavalla, eli pelimekaniikkoja ja ominaisuuksia muokattiin jatkuvasti ilman kaiken kattavaa pitkäaikaissuunnitelmaa. Pelin suunnittelu alkoi ideasta tehdä prototyyppi, jossa voidaan ajaa hyvin nopeasti mäkisessä maastossa, mikä antaisi mahdollisuuden tehdä suuria hyppyjä ja vauhdin tuntua. Pisteidenkeräysmekaniikkaa en ollut suunnitellut kovin tarkkaan, sillä minulla oli useita mahdollisia eri mekaniikkoja mielessä.

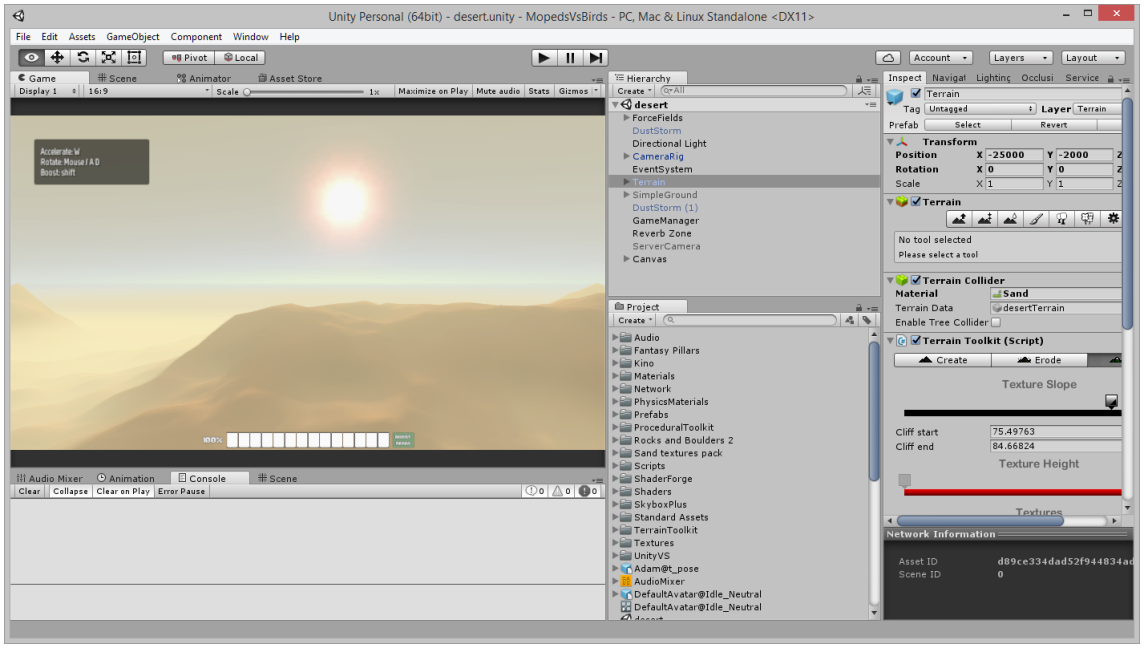
2 KÄYTETYT TYÖKALUT

2.1 Unity

Unity on Unity Technologiesin kehittämä monipuolinen pelimoottori ja pelinkehitystyökalu. Tärkeimmät Unityn ominaisuudet ovat sen pelimoottori ja editori sekä tuki yhteensä 23 eri pelialustalle. Tuetuista alustoista suosituimmat ovat Microsoft Windows, mobiililaitteiden Googlen Android ja Applen iOS, sekä konsolit Playstation 4 ja Xbox One. (Unity 2016a, viitattu 31.8.2016.)

Unityn editori on graafinen käyttöliittymä jolla voi hallita pelin objekteja ja ominaisuuksia erilaisista näkymistä. Näkymät ovat ikkunoita, joilla on omat tarkoituksensa editorissa. Editorin tärkeimmät ominaisuudet ovat jaettuna viiteen näkymään: Scene, Game, Inspector, Project sekä Hierarchy.

Scene-näkymä on visuaalinen näkymä pelimaailman muokkaamista varten. Game-näkymä näyttää pelin varsinaisen näkymän. Inspector-näkymä näyttää tietoa valittuna olevasta objektista ja mahdollistaa sen muokkauksen. Project-näkymä näyttää kaikki tiedostot projektin Assets-kansiosta. Hierarchy-näkymä näyttää kaikki pelin objektit nykyisessä scenessä. (Unity 2016b, viitattu 31.8.2016.)



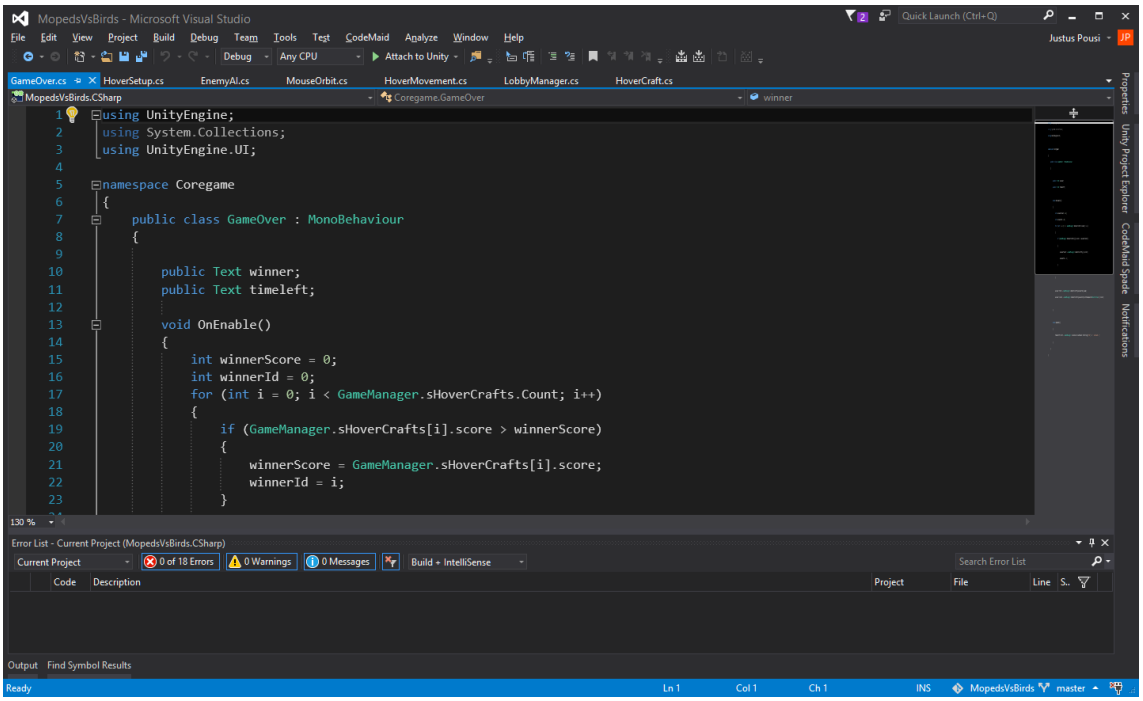
KUVIO 1. Unityn käyttöliittymä

Unitystä on eri versioita jotka ovat suunniteltu eri käyttäjäkunnille. Unity Personal on Unityn ilmainen versio minkä mukana tulee kaikki tärkeät ominaisuudet ja se sopii harrastelijoille sekä pienille startup-pelistudioille. Unity Plus sisältää hieman enemmän palveluita ja se on suunnattu askeleen isommille pelistudioille maksaen 35 dollaria kuukaudessa. Unity Pro on suunnattu isoille pelistudioille ja sisältää kaikki pilvipalvelut sekä maksaa 135 dollaria kuukaudessa. Unity Enterprise on suunnattu isoille AAA-pelistudioille ja sen hinta ja sisältämät palvelut ovat sopimuskohtaisia. (Unity 2016c, viitattu 31.8.2016.)

Unity Asset Store on kauppapaikka, josta voi hankkia Unityllä pelintekoon liittyvää sisältöä käytettäväksi omiin projekteihin. Asset Storessa on paljon sekä maksullista että ilmaista sisältöä, muun muassa 3D-malleja, animaatioita, esimerkkiprojekteja, tekstuureita, ääniä ja lisäosia Unity Editoriin. (Unity 2016d, viitattu 31.8.2016.)

2.2 Microsoft Visual Studio

Microsoft Visual Studio on ohjelmankehitysympäristö Microsoftilta. Sitä käytetään tietokoneohjelmien tekemiseen Microsoft Windowsille, kuten myös verkkosivuja ja verkko-ohjelmia ja palveluita. Visual Studio tukee monia ohjelmointikieliä kuten C#, Visual Basic, F#, C++, JavaScript, TypeScript ja Python. (Microsoft 2016a, viitattu 1.9.2016.)



KUVIO 2. Visual Studion käyttöliittymä

Unityllä on myös tuki Visual Studiolle Visual Studio Tools for Unity (VSTU) -pakkauksen avulla (Microsoft 2016b, viitattu 1.9.2016). Visual Studion avulla voi kirjoittaa ja muokata C# -koodia helposti IntelliSense-ominaisuuden avulla. Visual Studio tukee myös Unityssä virheiden etsintää käymällä koodia rivi kerrallaan samalla kun peli on käynnissä. (Microsoft 2016c, viitattu 1.9.2016.)

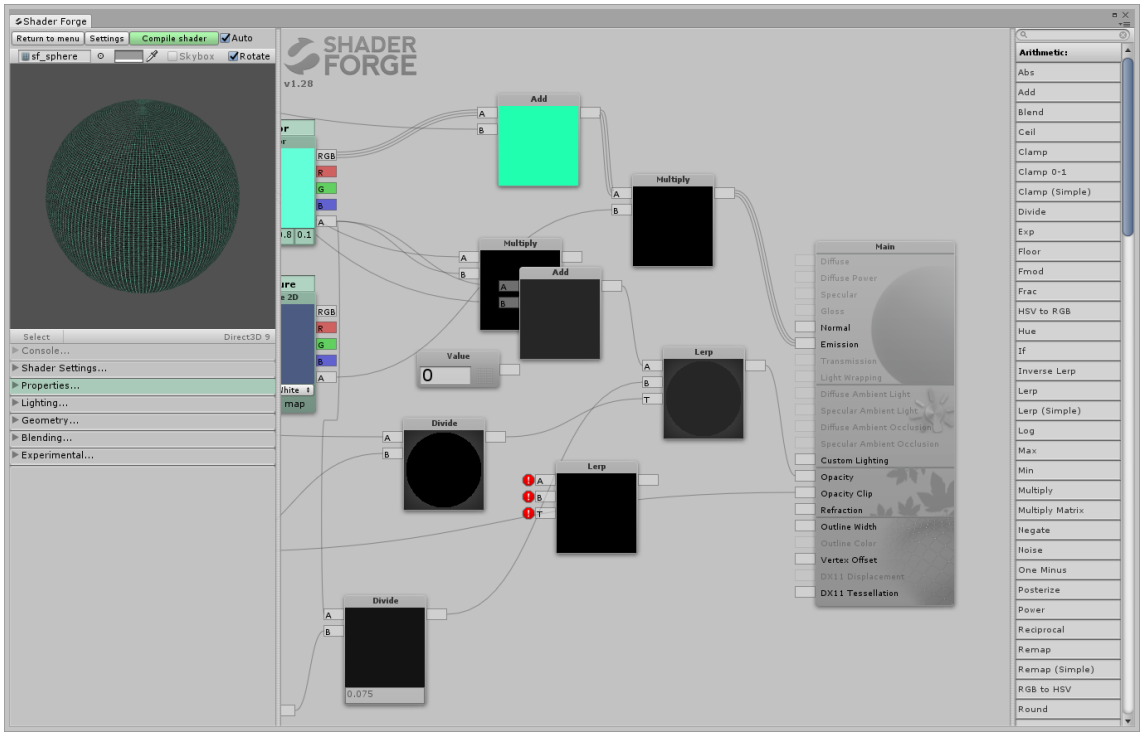
2.3 Blender

Blender on 3D-grafiikan mallinnusohjelma. Sitä levitetään ilmaisen GPL-lisenssin alla (Blender Foundation 2016a, viitattu 31.8.2016). Blender on suosittu sekä ammattilaisten että harrastelijoiden käytössä. Blenderillä on kattavat ominaisuudet tehdä 3D-mallinnusta ja animaatioita elokuviin, peleihin sekä muihin käyttötarkoituksiin (Blender Foundation 2016b, viitattu 31.8.2016). Blenderillä on virallisesti julkaistu Windowsille, Mac OS X:lle sekä Linuxille 32-bittisenä sekä 64-bittisinä versioina (Blender Foundation 2016c, viitattu 31.8.2016).

Unity tukee automaattisesti Blenderin tiedostoja käyttäen Blender FBX exporter -ominaisuutta. Jotta Blender objektin saa vietyä Unityyn, täytyy .blend tiedosto tallentaa Unity projektin Assets kansioon. Kun Unityyn menee takaisin, tiedosto automaattisesti tuodaan Unityyn ja sen pitäisi näkyä Unityn projektinäkymässä. (Unity 2016e, viitattu 1.9.2016.)

2.4 Shader Forge

Shader Forge on Unityssä toimiva graafinen editori shader-skriptien luomiseen. Shader-skripteillä voi muokata peliobjektien ulkonäköä ja tehdä visuaalisia efektejä ilman tarvetta ohjelmoida. Ne ovat vastuussa väreistä jotka objektin pinnalla näytetään. Värit lasketaan käyttäen tietoa kuten tekstuurit, valon lähteet, valot värit ja objektin pinnan tiedot. (Neat Corporation 2015, 1.)



KUVIO 3. Shader Forgen käyttöliittymä

Shader Forgen on luonut Joachim Holmér ja se on maksullinen Unity Asset Storessa. Shader Forgea ovat kokeilleet useat pelinkehittäjät yrityksistä kuten Valve, Blizzard, DICE, Ubisoft, Bungie, EA, Avalanche Studios ja Rovio. (Unity 2016f, viitattu 1.9.2016.)

3 UNITYN HLAPI-RAJAPINTA

The High Level API (HLAPI) eli korkean tason rajapinta on Unityn 5.1 version mukana julkaistu järjestelmä jolla voidaan rakentaa moninpeliominaisuuksia Unity peleihin. HLAPI kutsutaan joskus myös nimellä UNET (Unity Networking). HLAPI on rakennettu matalan tason reaaliaikaisen kommunikaatiotason (LLAPI) päälle helpottamaan monien yleisten tehtävien hallintaa verkkopeleissä. (Lian 2015, viitattu 31.8.2016.)

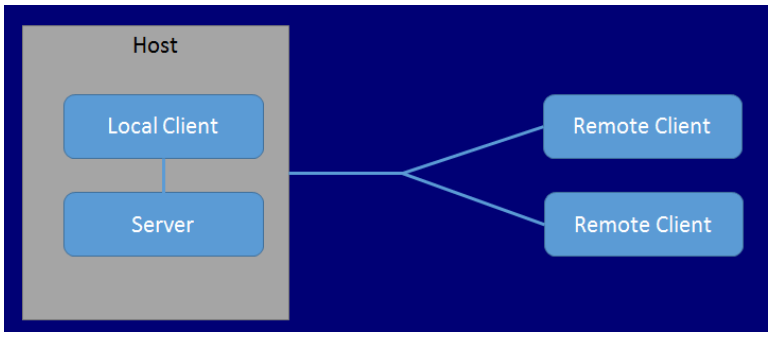
Korkean tason rajapinnalla tarkoitetaan, että se tarjoaa ominaisuuksia jotka ovat enemmän abstrakteja kuin alemman tason rajapinnalla. Korkealla tasolla tavoitteet ja ominaisuudet koskevat tyypillisesti järjestelmän kokonaisuutta laajemmin. Abstraktion tarkoituksena on säilyttää kontekstiin liittyvä tärkeä tieto ja unohtaa kontekstiin merkityksetön tieto. (Guttag 2013, 43.)

HLAPI on palvelin-auktoriteettinen järjestelmä. Sen avulla myös yksi päätteistä voi olla palvelin samalla aikaa, jolloin ei tarvita erillistä pelille omistettua palvelinta. Tämä mahdollistaa verkkopelien pelaamisen internetin yli vähällä kehittämisvaivalla. (Unity 2016g, viitattu 22.9.2016.)

3.1 Palvelin ja isäntä

Unityn verkkojärjestelmässä, peleillä on palvelin (server) ja useita päätteitä (client). Silloin kun ei ole olemassa pelille omistettua palvelinta, yksi päätteistä ottaa palvelimen roolin. Tätä päätettä kutsutaan isännäksi (host). (Unity 2016m, viitattu 22.9.2016.)

Isäntä on palvelin ja pääteohjelma yhtä aikaa. Isäntä käyttää erityistä paikallista pääteohjelmaa “LocalClient”, kun taas muut päätteet ovat etäpäätteitä eli “RemoteClient”. LocalClient on suorassa yhteydessä paikallisen palvelimen kanssa, sillä ne ovat osa samaa prosessia. RemoteClient on yhteydessä palvelimen kanssa tavallisen verkkoyhteyden kautta. (sama)



KUVIO 4. Isäntäpalvelin ja kaksi etäpäätettä

3.2 HLAPI-komponentit ja luokat

HLAPI sisältää kokoelman Unity-komponentteja ja ohjelmoinnissa käytettäviä luokkia, joilla pystyy editorissa luomaan monet tärkeimmät verkkopelien ominaisuudet. HLAPI-komponentteja lisätään editorissa peliobjekteihin AddComponents-valikosta. Tässä on lista pelin prototyypissä käyttämistäni HLAPI-komponenteista.

3.2.1 NetworkIdentity

NetworkIdentity-komponentti on Unityn verkkojärjestelmän ytimenä. Tämä komponentti kontrolloi objektin verkkoidentiteetin, ja se saa verkkojärjestelmän olemaan tietoinen siitä. (Unity 2016h, viitattu 22.9.2016.)

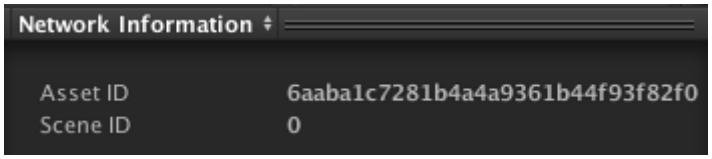
Pelaajaobjektille voidaan asettaa paikallinen auktoriteetti (local authority). Tämä tarkoittaa, että pelaajaobjekti on sen omalla päätteellä vastuussa objektista. Tätä käytetään useimmiten liikkumisen kontrollointiin, mutta voidaan käyttää myös muissa asioissa. NetworkTransform-komponentti ymmärtää tämän ja lähettää liikkumisen päätteeltä, jos tämä on asetettu. (sama)

KUVIO 5. Tältä näyttää NetworkIdentity-komponentti

Scenessä olevia objekteja kohdellaan eri tavalla kuin verkkoon dynaamisesti luotuja objekteja. Kun peli käynnistetään, kaikki objektit joilla on NetworkIdentity-komponentti, otetaan automaattisesti pois päältä. Kun pääte yhdistää palvelimeen niin palvelin kertoo päätteelle mitkä objektit kuuluvat olla päällä ja mikä niiden viimeisin tilanne kuuluu olla. Tämä takaa, että kun

pelaaja aloittaa pelaamaan niin hänen päätteensä asettaa kaikki objektit oikeille paikoille, ja ei luo objekteja jotka kuuluisi olla poistettuna. (Unity 2016h, viitattu 22.9.2016.)

Jotta samanlaiset objektit voidaan tunnistaa toisistaan, NetworkIdentity-komponentti sisältää seurantatietoa kuten scene-tunnisteen (sceneID), verkkotunnisteen (networkID) ja asset-tunnisteen (assetID). (sama)



KUVIO 6. NetworkIdentity-komponentin seurantatietoja

3.2.2 NetworkBehaviour

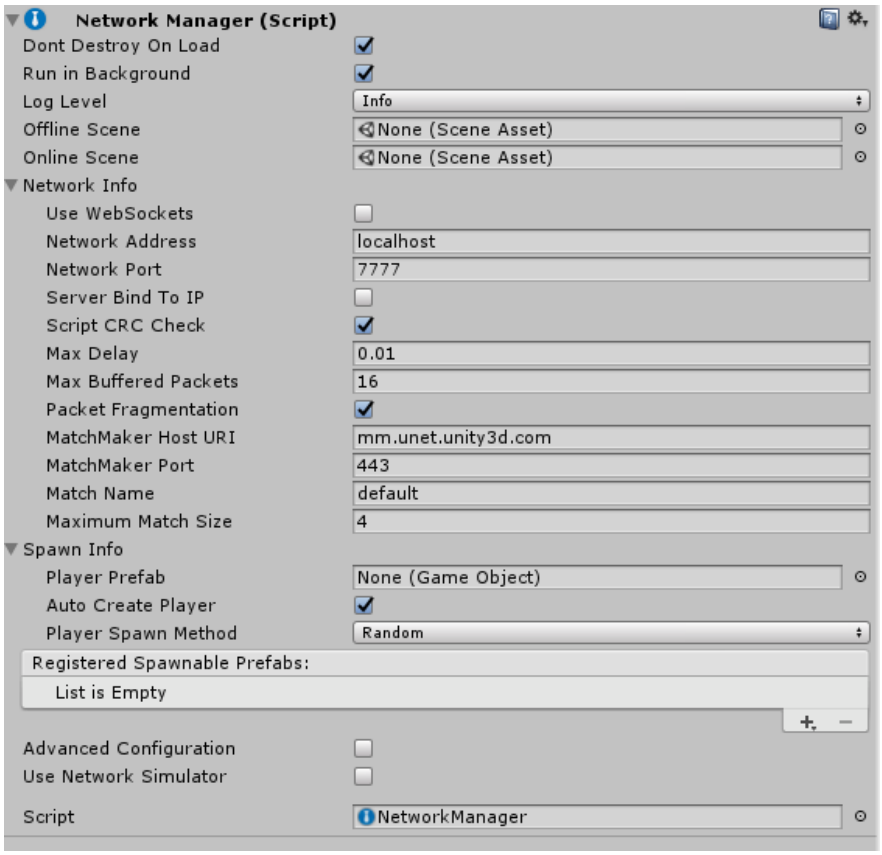
NetworkBehaviour on ohjelmoinnissa käytettävä luokka (class), jota tarvitaan HLAPI skripteissä. Skriptit jotka perivät NetworkBehaviour-luokan voidaan asettaa peliobjektiin AddComponents-valikosta, ja ne vaativat NetworkIdentity-komponentin. Nämä skriptit voivat suorittaa HLAPI funktioita kuten Command, ClientRPC, SyncEvent, ja SyncVar. (Unity 2016i, viitattu 22.9.2016.)

```
public class HoverCraft : NetworkBehaviour
{
```

KUVIO 7. Esimerkki luokasta joka perii NetworkBehaviour-luokan

3.2.3 NetworkManager

NetworkManager-komponentti on verkkojärjestelmien käyttöä helpottava luokka. Sen avulla voidaan hallita päätteiden ja palvelimien yhteyksiä. Edistyneet ohjelmoijat voivat laajentaa NetworkManager-luokan toimintaa virtuaalifunktiolla. (Unity 2016j, viitattu 22.9.2016.)



KUVIO 8. NetworkManager-komponentin käyttöliittymä

NetworkManagerHUD-komponentti on pieni lisäosa NetworkManager-komponenttiin. Se näyttää pelissä yksinkertaisen käyttöliittymän joka avulla voidaan ohjata verkkoyhteyksiä pelissä. Tämä on hyödyksi verkkoprojektin opetteluun, mutta ei ole tarkoitettu käytettäväksi lopulliseen peliin. (Unity 2016j, viitattu 22.9.2016.)

3.2.4 NetworkLobbyManager

NetworkLobbyManager on laajennettu versio NetworkManager-luokasta, johon on lisätty aula (lobby). Aula on eräänlainen valikko johon pelaajat liittyvät ennen pelin alkamista ja jossa pelaajat voivat nähdä pelin asetukset. (Unity 2016m, viitattu 22.9.2016.)

3.2.5 NetworkTransform

NetworkTransform-komponentti synkronoi objektin paikan ja rotaation pelimaailmassa muille pelaajille. Liikkuvat objektit tarvitsevat tätä verkkopeleissä. Jos objektin NetworkIdentityllä on paikallinen auktoriteetti, sen liikkeet synkronoidaan päätteeltä palvelimelle ja sieltä muille päätteille. Jos palvelimella on auktoriteetti niin palvelin kontrolloi liikkeitä ja lähettää ne suoraan kaikille päätteille. (Unity 2016k, viitattu 22.9.2016.)

3.3 Synkronoidut muuttujat

SyncVar on NetworkBehaviour-luokan muuttuja, jota käytetään merkkaamaan muuttujat synkronoitavaksi. Mikä tahansa perustyyppinen muuttuja tai Unityn oma tietotyyppi voi olla SyncVar, paitsi ei luokat, listat tai muut kokoelmat. Kun SyncVar muuttujan arvo muuttuu, se lähetetään kaikille pelin päätteille päivitettäväksi. Kun verkkopelissä luodaan uusi objekti, ne luodaan päätteelle viimeisimpien palvelimelta saatujen SyncVar arvojen kanssa. Muuttujasta tehdään SyncVar muuttuja laittamalla sen eteen [SyncVar]-attribuutin. (Unity 2016l, viitattu 22.9.2016.)

```
public class SpaceShip : NetworkBehaviour
{
    [SyncVar]
    public int health;

    [SyncVar]
    public string playerName;
}
```

KUVIO 9. SyncVar esimerkki

SyncList on kuin SyncVar mutta se on lista arvoista yksittäisen arvon sijaan. SyncList on oma luokkansa, joten se ei tarvitse SyncVar-attribuuttia toimiakseen. Sisäänrakennettuja SyncList-tyyppejä on viisi erilaista: SyncListString, SyncListFloat, SyncListUInt sekä SyncListBool. (Unity 2016l, viitattu 22.9.2016.)

3.4 Verkon callback-funktiot

NetworkBehaviour-luokassa on callback-funktioita, joita kutsutaan tietyille verkon tapahtumille. Ne ovat virtuaalifunktioita, joten niiden päälle voidaan kirjoittaa ja niiden ominaisuuksia voidaan laajentaa. Callback-funktioita voidaan esimerkiksi käyttää tekemään asioita joita pitää rajoittaa joko palvelimelle tai päätteelle. (Unity 2016i, viitattu 22.9.2016.)

```
public class SpaceShip : NetworkBehaviour
{
    public override void OnStartServer()
    {
        // disable client stuff
    }

    public override void OnStartClient()
    {
        // register client events, enable effects
    }
}
```

KUVIO 10. Esimerkki callback-funktioista

3.5 Peliobjektien luominen verkossa

Unityssä funktio GameObject.Instantiate luo uusia peliobjekteja (GameObject) pelin sisälle. Jotta esineet luodaan kaikille päätteille, ne täytyy luoda myös koko aktiiviselle verkolle. Tämä voidaan tehdä vain palvelimella, ja aiheuttaa esineiden luonnin kaikilla yhdistetyillä pääteohjelmilla. (Unity 2016d, viitattu 22.9.2016.)

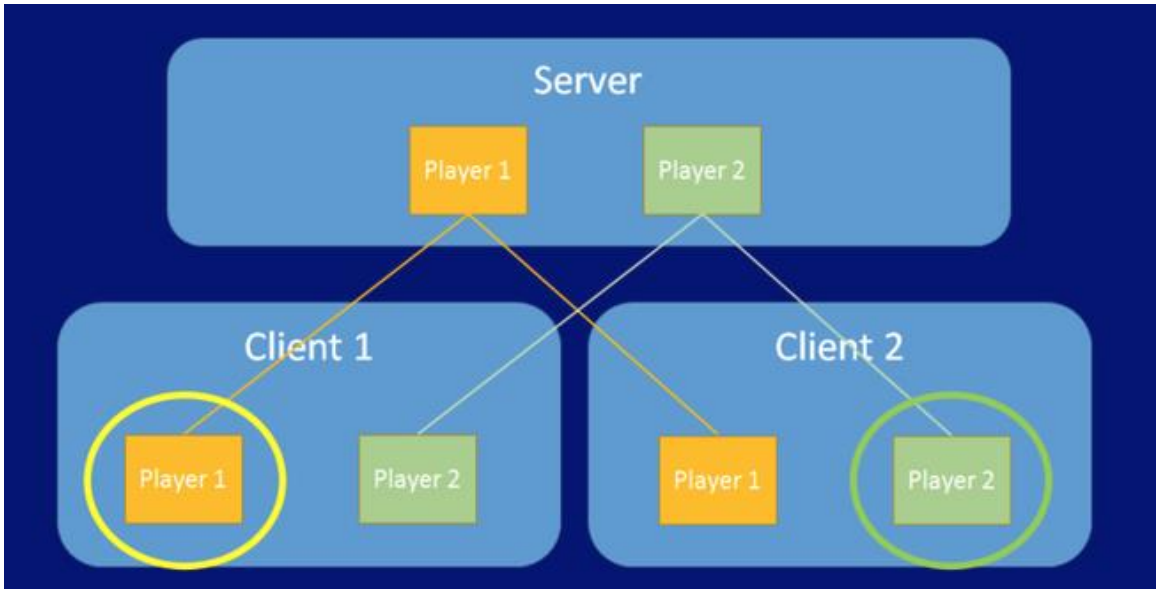
Objektit joilla on NetworkIdentity-komponentti, täytyy luoda verkkoon (spawn) palvelimella käyttäen NetworkServer.Spawn() funktiota. Tällöin niille automaattisesti annetaan verkkotunnus (NetworkInstanceId) ja ne luodaan kaikille palvelimelle yhteydessä oleville päätteille. (Unity 2016h, viitattu 22.9.2016.)

```
class Tree : NetworkBehaviour
{
    [SyncVar]
    public int numLeaves;
}

class MySpawner : NetworkBehaviour
{
    public GameObject treePrefab;

    public void Spawn()
    {
        GameObject tree = (GameObject)Instantiate(treePrefab, transform.position, transform.rotation);
        tree.GetComponent<Tree>().numLeaves = Random.Range(10,200);
        NetworkServer.Spawn(tree);
    }
}
```

KUVIO 11. Esimerkki objektin luomisesta verkossa

3.6 Verkkojärjestelmän auktoriteetti

Verkkopelissä on olemassa konsepti “omasta” pelaajaobjektista jonka päätteet osaavat tunnistaa. Jokaisella pelaajalla on oma pelihahmo jota vain pelaaja itse voi ohjata. Tätä objektia kutsutaan paikalliseksi pelaajaksi (local player) ja sillä on attribuutti “isLocalPlayer” joka asetetaan todeksi. Tätä voidaan käyttää suodattamaan kontrollisyötettä, käsittelemään kameran liitäntää, tai mitä tahansa muuta pääteohjelman tapahtumia jotka kuuluvat tapahtua vain omalla pelaajalla. Ei-pelaaja objekteilla kuten vihollisilla ei ole omaa päätettä, joten niiden auktoriteetti kuuluu palvelimelle. (Unity 2016m, viitattu 22.9.2016.)



KUVIO 12. Kaksi pääteohjelmaa ja ympyröitynä niiden paikalliset pelaajat

3.7 Etätoiminnot

Unityn verkkojärjestelmässä on useita tapoja suorittaa toimintoja verkon yli. Tämän tyyppisiä toimintoja kutsutaan etäproseduurikutsuiksi eli “Remote Procedure Calls” (RPC). HLAPI käyttää kahdenlaisia RPC-komentoja: Command joka kutsutaan päätteeltä ja suoritetaan palvelimella; sekä ClientRPC joka kutsutaan palvelimella ja suoritetaan päätteellä. (Unity 2016i, viitattu 22.9.2016.)

Command-kutsu lähetetään päätteen pelaajaobjektista palvelimen pelaajaobjektille. Tietoturvan vuoksi Command-kutsu voidaan lähettää vain omasta pelaajaobjektista, joten pelaajat eivät voi muokata muiden pelaajien peliobjekteja. Funktiosta tehdään Command-kutsu lisäämällä [Command]-attribuutti funktion eteen. Lisäksi funktiolla täytyy olla Cmd-etuliite nimessä. Tämä antaa vinkin siitä, että tämä funktio on erilainen ja sitä ei kutsuta kuten tavallista funktiota. (Unity 2016i, viitattu 22.9.2016.)

```
class Player : NetworkBehaviour
{

    public GameObject bulletPrefab;

    [Command]
    void CmdDoFire(float lifeTime)
    {
        GameObject bullet = (GameObject)Instantiate(
            bulletPrefab,
            transform.position + transform.right,
            Quaternion.identity);

        var bullet2D = bullet.GetComponent<Rigidbody2D>();
        bullet2D.velocity = transform.right * bulletSpeed;
        Destroy(bullet, lifeTime);

        NetworkServer.Spawn(bullet);
    }

    void Update()
    {
        if (!isLocalPlayer)
            return;

        if (Input.GetKeyDown(KeyCode.Space))
        {
            CmdDoFire(3.0f);
        }

    }
}
```

KUVIO 13. Esimerkki Command-etätoiminnon käytöstä

ClientRPC-kutsut lähetetään objektista serveriltä niitä vastaaviin objekteihin päätteille. Niitä voidaan lähettää miltä tahansa palvelimen objektista jolla on NetworkIdentity-komponentti ja jotka on luotu verkossa. Funktiosta tehdään ClientRPC-kutsu lisäämällä [ClientRPC]-attribuutti ennen funktiota, ja lisäämällä funktion nimeen Rpc-etuliite. (Unity 2016i, viitattu 22.9.2016.)

```
using UnityEngine;
using UnityEngine.Networking;

public class SpaceShipRpc : NetworkBehaviour
{
    [ClientRpc]
    public void RpcDoOnClient(int foo)
    {
        Debug.Log("OnClient " + foo);
    }

    [ServerCallback]
    void Update()
    {
        int value = UnityEngine.Random.Range(0,100);
        if (value < 10)
        {
            // this will be invoked on all clients
            RpcDoOnClient(value);
        }
    }
}
```

KUVIO 14. Esimerkki ClientRPC-etätoiminnon käytöstä

3.8 HLAPI verrattuna muihin Unityn verkkorajapintoihin

Unityn Transport Layer-rajapinta on matalan tason rajapinta, jota kutsutaan joskus nimellä LLAPI (low-level API). Se tarjoaa mahdollisuuden rakentaa oman verkkojärjestelmän peliin, mikä on hyödyllistä, jos tarvitaan jotain tiettyjä tai edistyksellisiä vaatimuksia pelin verkossa. Korkeamman tason abstraktion takia HLAPI on helpompi ottaa käyttöön kuin Transport Layer-rajapinta. Transport Layer-rajapintaa voidaan myös käyttää yhdessä HLAPI:n kanssa. (Unity 2016p, viitattu 2.10.2016.)

Photon Unity Networking (PUN) on Exit Games yhtiön tarjoama alusta verkkopelien tekemiseen Unityllä. PUN mukana tuleva rajapinta on hyvin samankaltainen kuin Unityn HLAPI. PUN käyttää Photon Cloud-palvelun palvelimia verkkopelien ylläpitämiseen. PUN tarjoaa ilmaisen mutta rajatun paketin käyttäjille Unity Asset Storessa. Siitä on myös tällä hetkellä 95 dollaria maksava

versio joka sisältää laajemman tuen eri alustoille, sekä suuremman käyttäjäkapasiteetin Photon Cloud palvelimille. (Exit Games 2016, viitattu 2.10.2016.)

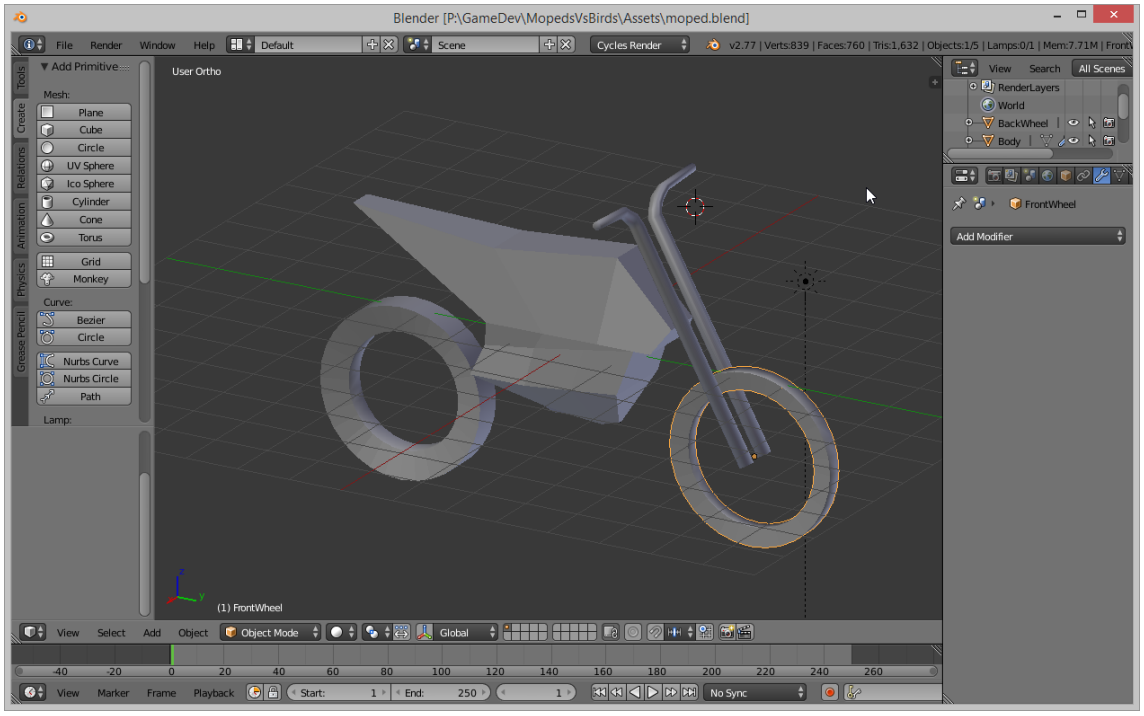
Yksi suurimmista eroista HLAPI ja PUN välillä on, että PUN käyttämät Photon Cloud palvelimet eivät ole auktoritaarisia. Tämä tarkoittaa, että palvelimella ei voida ajaa omia skriptejä, ja että huijareiden ennaltaehkäiseminen on lähes mahdotonta. (Ignatchenko 2016, viitattu 2.10.2016.)

4 PELIN PROTOTYYPPI

Prototyypin tavoitteena oli kokeilla miettimääni pelimekaniikkaa sekä opetella käyttämään Unityn HLAPI-verkkojärjestelmää. Työskentelin yksin projektin parissa satunnaisesti noin vuoden ajan. Satunnaiset ongelmat veivät paljon aikaa ja motivaatiota projektista.

Alkuperäinen idea pelissä oli käyttää nopeaa ja taidokasta liikkumista pelimekaniikkana kukkulaisessa maastossa. Suurin inspiraatio ideaan oli Tribes Ascend –peli jossa pelaaja voi saavuttaa hyvin nopean vauhdin käyttämällä pelin ”hiihto”-mekaniikan erikoista fysiikkaa.

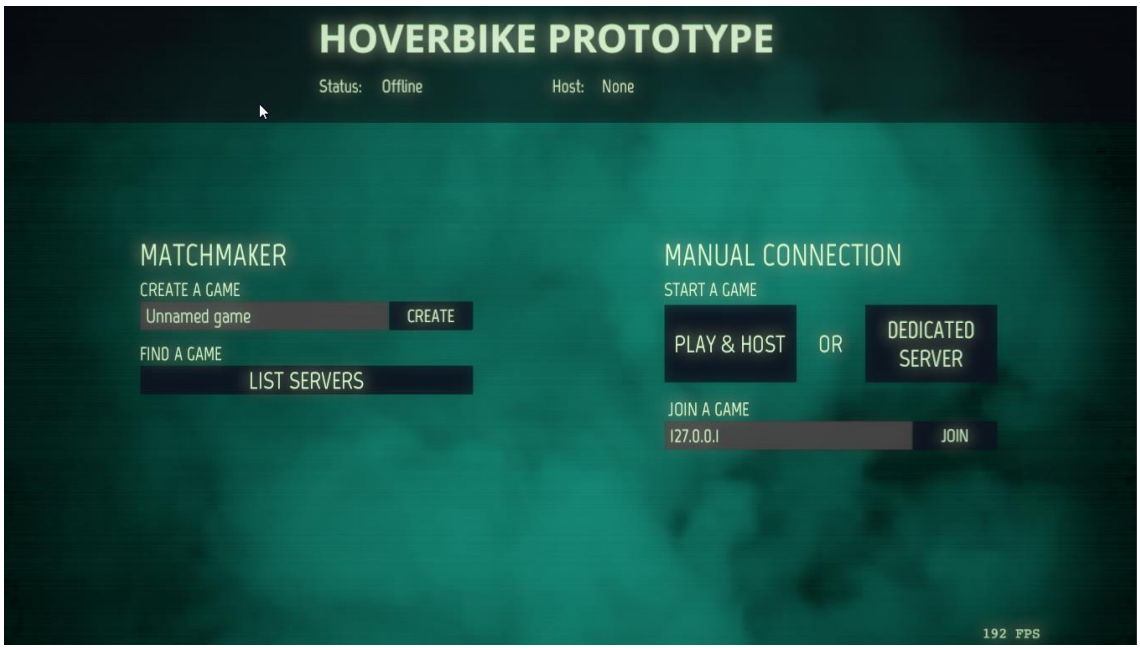
Aloitin pelin tekemisen, luomalla Blender-mallinnusohjelmalla yksinkertaisen 3d-mallin crossipyörästä. Sain crossipyörän toimimaan Unityssä käyttämällä WheelCollider-komponenttia. WheelCollider on suunniteltu autoja varten, joten moottoripyörällä se oli liian hankala saada käyttäytymään realistisesti. Sain sitten idean tehdä crossipyörästä leijuvan ”hovercraftin”, jotta minun ei tarvitsisi huolehtia ajoneuvon realistisesta liikkumisesta. Hovercraftin tein käyttämällä Unityn sisäänrakennettua PhysX-fysiikkamoottoria ajoneuvon leijumiseen, liikkumiseen ja kääntymiseen. Hovercraftin kontrolloinnin muokkaaminen sopivaksi oli myös aikaa vievää, ja vaati paljon erilaisten ratkaisujen kokeilemista.



KUVIO 15. Crossipyörä Blender-mallinnusohjelmassa

Aluksi tein peliin maaston Blenderillä. Tein muutamia suuria hiekkadyynejä kokeiluksi, joissa pystyi ottamaan paljon vauhtia ja tekemään isoja hyppyjä. Halusin kuitenkin kokeilla miltä peli tuntuisi proseduurillisesti luodulla maastolla, joten otin käyttöön MIT lisenssillä julkaistun Terrain Toolkitin. Tällä pystyin korvaamaan vanhan staattisen maaston semmoisella, joka on erilainen jokaisella pelikerralla.

Moninpelin saaminen sulavaksi oli myös alussa hankalaa. Alun perin kokeilin tehdä oman skriptin, jolla synkronoin monipelissä pelaajien liikkeet. Se kuitenkin osoittautui liian hankalaksi, joten päädyin käyttämään NetworkTransform-komponenttia joka synkronoi liikkeet automaattisesti. NetworkTransform-komponentin liikkeiden synkronoinnin sulavuudessa on kuitenkin vielä joitain ongelmia. Kokeilin myös tehdä oman NetworkManager-luokkaan perustuvan verkonhallinta-komponentin, joka osoittautui myös liian hankalaksi ja aikaa vieväksi. Päädyin käyttämään Network Lobby -paketissa olevaa ratkaisua. Siinä tulee toimiva aula, jonka avulla pelaaja voi luoda uuden pelin tai liittyä olemassa olevaan peliin. Muokkasin aulan ulkonäköä itselle mieluisaksi. Käytin Shader Forgea taustalla olevan efektin tekemisessä.



KUVIO 16. Pelin aula

Pelin maaston luomiseen käytettiin TerrainGenerator-skriptiä. Pelin alkaessa TerrainGenerator-skriptin maaston luomiseen käytettävät arvot arvotaan palvelimella, jotta maastosta tulisi erilainen joka pelikerralla. Tämän jälkeen maasto luodaan kaikilla päätteillä, ja palvelin luo maaston päälle satunnaisia esineitä. NetworkBehaviour-luokan isServer-arvo oli hyödyllinen, kun komentoja pitää ajaa pelkästään palvelimella (katso kuvio 17).

```
void Start()
{
    TerrainGenerator tg = FindObjectOfType<TerrainGenerator>();

    // Tell the server to randomize the terrain.
    if (isServer)
        tg.RandomizeValues();

    // Generate the terrain for all clients
    tg.Generate();

    // Start the fuel spawning coroutine and spawn debris
    if (isServer)
    {
        StartCoroutine(FuelSpawnCoroutine());
        SpawnObject("pillars", pillarPrefab, 20, -Random.Range(5f, 20f), true);
        SpawnObject("rocks", rockPrefab, 100, 0, true);
        SpawnObject("enemies", enemyPrefab, 50, 20f, false);
    }
```

KUVIO 17. Esimerkki NetworkBehaviour-luokan isServer-arvosta

Kun pelitilanne muuttuu, esimerkiksi pelin loputtua, palvelimen täytyy viestittää muutos kaikille päätteille. Tähän käytetään ClientRPC-etätoimintoja (katso kuvio 18).

```
[ClientRpc]
void RpcGameOverScreen()
{
    _gameOverScreen.SetActive(true);
    Time.timeScale = 0.3f;
    Time.fixedDeltaTime = 0.005f;

}

[ClientRpc]
void RpcResetTime()
{
    Time.timeScale = 1f;
    Time.fixedDeltaTime = 0.01f;

}
[ClientRpc]
void RpcUnpauseEveryone()
{
    LobbyManager.s_Singleton.loadingscreen.SetActive(false);
    Time.timeScale = 1f;
}
```

KUVIO 18. Pelin keskeytystä käsittelevät RPC-funktiot

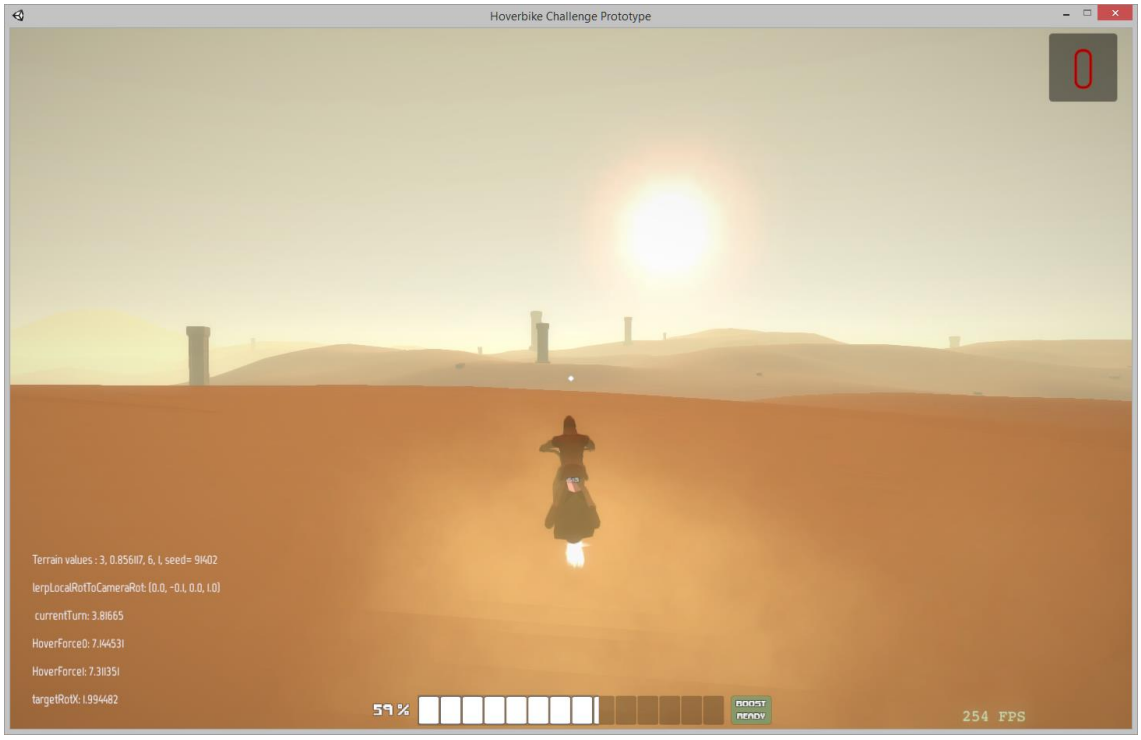
Kun peli on päättynyt, alkaa ajastin jonka jälkeen peli palaa takaisin aulaan. Jotta ajastin saadaan synkronoitua kaikkien päätteiden välillä, täytyy käyttää liukulukumuuttujaa, joka on asetettu SyncVar-muuttujaksi (katso kuvio 19). Ajastimen arvo pienenee, kunnes arvo on pienempi kuin nolla, jolloin peli palaa takaisin aulaan.

```
[SyncVar]
public float endTimer = 4f;

IEnumerator ReturnToLobby()
{
    // Calling game over menu here
    RpcGameOverScreen();

    while (endTimer > 0f)
    {
        endTimer -= Time.unscaledDeltaTime;
        yield return null;
    }
    RpcResetTime();

    LobbyManager.s_Singleton.ServerReturnToLobby();
}
```

KUVIO 19. Synkronoitu muuttuja joka toimii ajastimena

Unityn NetworkLobbyManager-komponentti oli hyödyllinen pelien moninpeli prototyypin kehittelyyn, mutta sen funktiot ovat liian suljettuja ja huonosti dokumentoituja, jotta sitä kannattaisi käyttää lopullisen pelin kehittämisessä.

Pelin varsinainen päämäärä on yksinkertainen. Peliä voi pelata yksin tai moninpelinä. Pelissä jokaisella pelaajalla on hovercraft-ajoneuvo, jolla on rajallinen määrä polttoainetta. Pelaajan tehtävänä on kerätä sinisiä hohtavia palloja, joista saa lisää polttoainetta ja lisää pisteitä. Palloja ilmestyy pelikentälle satunnaisiin paikkoihin tietyin ajanvälein. Peli loppuu, kun kaikilta pelaajilta on loppunut polttoaine.



KUVIO 20. Esimerkki pelitilanteesta

5 POHDINTA

Työn tavoitteena oli tutustua Unityn HLAPI-rajapintaan ja tehdä verkkopelin prototyyppi sitä käyttämällä. Nämä tavoitteet onnistuivat hyvin, vaikka itse peli on vielä kesken. Suurin puute pelissä tällä hetkellä on, että peli ei ole kovin hauska, sillä en suunnitellut pelin pelimekaanikkoja alusta asti kovin hyvin.

Kun aloitin työskentelemään raporttia, suurin ongelma joka minulla oli, että kuinka rajaan aiheen tarpeeksi hyvin. Aluksi ajattelin, että kirjoitan pelkästä prototyypin tekemisestä, mutta kun aloin hahmottelemaan raportin runkoa, tajusin että on helpompi kirjoittaa hyödyllistä sisältöä, jos aihe on tarkemmin rajattuna. Päätin sen jälkeen rajata kirjoittamaan vain prototyypin verkkojärjestelmästä ja jättää pelin suunnittelu sekä graafinen ulkoasu pääosin pois raportista.

Aiemmat kokemukseni verkkopelien kehittämisestä olivat hyvin vähäiset. Ennen prototyypin aloitusta, olin suorittanut joitain YouTubessa olevia opetusohjelmia Photon Unity Networking -paketin kanssa. Prototyyppiä ja raporttia tehdessä opin paljon enemmän yleisesti verkkopelien tekemisestä. Unityn HLAPI-järjestelmä tarjoaa hyvät työkalut moninpelin tekemiseen, ja se automatisoi matalan tason verkkoliikennettä, jotta kehittäjän ei tarvitse niistä huolehtia. HLAPI:n dokumentointi on melko hyvä Unityn omassa manuaalissa, mutta parannettavaa olisi varsinkin NetworkManager ja NetworkLobbyManager -komponenttien osalta.

Pidän mahdollisena suunnitelmana, että jatkan pelin kehitystä jossain vaiheessa. Peliin pitäisi ensin suunnitella hauskempi pelimekaniikka kuin nykyinen polttoaineen ja pisteiden keräys. Maaston luontia pitäisi myös parantaa. Nykyinen Terrain Toolkit jolla luon maaston, on väliaikainen ratkaisu. Suunnitelmana on tehdä itse skriptin maaston proseduurilliseen luomiseen, jolla korvaan nykyisen ratkaisun.

LÄHTEET

Lian, A. 2015. Unity 5.1 is here! Viitattu 31.8.2016, https://blogs.unity3d.com/2015/06/09/unity-5-1-is-here/.

Unity Technologies 2016a. Build once deploy anywhere. Viitattu 31.8.2016, https://unity3d.com/unity/multiplatform.

Unity Technologies 2016b. Interface & essentials. Viitattu 31.8.2016, https://unity3d.com/learn/tutorials/topics/interface-essentials.

Unity Technologies 2016c. Asset Store Help. Viitattu 31.8.2016, https://unity3d.com/asset-store/help.

Unity Technologies 2016d. Asset Store. Viitattu 31.8.2016, https://www.assetstore.unity3d.com/en/.

Unity Technologies 2016e. Importing Objects From Blender. Viitattu 1.9.2016, https://docs.unity3d.com/Manual/HOWTO-ImportObjectBlender.html.

Unity Technologies 2016f. Shader Forge. Viitattu 1.9.2016, https://www.assetstore.unity3d.com/en/#!/content/14147.

Unity Technologies 2016g. The High Level API. Viitattu 22.9.2016, https://docs.unity3d.com/Manual/UNetUsingHLAPI.html.

Unity Technologies 2016h. NetworkIdentity. Viitattu 22.9.2016, https://docs.unity3d.com/ScriptReference/Networking.NetworkIdentity.html.

Unity Technologies 2016i. NetworkBehaviour. Viitattu 22.9.2016, https://docs.unity3d.com/ScriptReference/Networking.NetworkBehaviour.html.

Unity Technologies 2016j. NetworkManager. Viitattu 22.9.2016, https://docs.unity3d.com/Manual/UNetManager.html.

Unity Technologies 2016k. NetworkTransform. Viitattu 22.9.2016, https://docs.unity3d.com/ScriptReference/Networking.NetworkTransform.html.

Unity Technologies 2016l. State Synchronization. Viitattu 22.9.2016, https://docs.unity3d.com/Manual/UNetStateSync.html.

Unity Technologies 2016m. Network Lobby Manager. Viitattu 22.9.2016, http://docs.unity3d.com/Manual/UNetConcepts.html.

Unity Technologies 2016n. Network System Concepts. Viitattu 22.9.2016, https://docs.unity3d.com/Manual/class-NetworkLobbyManager.html.

Unity Technologies 2016o. Network System Concepts. Viitattu 22.9.2016, https://www.assetstore.unity3d.com/en/#!/content/41836.

Unity Technologies 2016p. Using the Transport Layer API. Viitattu 2.10.2016, https://docs.unity3d.com/Manual/UNetUsingTransport.html.

Blender Foundation 2016a. License. Viitattu 31.8.2016, https://www.blender.org/about/license/.

Blender Foundation 2016b. Features. Viitattu 31.8.2016, https://www.blender.org/features/.

Blender Foundation 2016c. Download Blender 2.77a. Viitattu 31.8.2016, https://www.blender.org/download/.

Neat Corporation 2015. Quick Start GUIDE. Viitattu 1.9.2016, http://www.neatcorporation.com/Projects/ShaderForge/Media/ShaderForge_QuickStart-en.pdf.

Microsoft 2016a. Visual Studio Community. Viitattu 1.9.2016, https://beta.visualstudio.com/vs/community/.

Microsoft 2016b. Getting Started with Visual Studio Tools for Unity. Viitattu 1.9.2016, https://msdn.microsoft.com/en-us/library/dn940025.aspx.

Microsoft 2016c. Build Unity Games with Visual Studio. Viitattu 1.9.2016, https://www.visualstudio.com/en-us/features/unitytools-vs.aspx.

Guttag, J. V. 2013. Introduction to Computation and Programming Using Python. Cambridge: MIT Press.

Exit Games 2016. Photon Unity Networking Intro. Viitattu 2.10.2016. http://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro.

Ignatchenko, S. 2016. Unity 5 vs UE4 vs Photon vs DIY for MMO. Viitattu 2.10.2016. http://ithare.com/unity-5-vs-ue4-vs-photon-vs-diy-for-mmo/.